ChatPub: Retrieval Augmented Generation-Based Web Application to Help Finding Policy for Korean Youth

Kangsan Kim(rkdtsk419@g.skku.edu), Seungbin Yang(didtmdqlsdl@gmail.com)
Jinho Park(jinho142857@gmail.com), Changmin Jun(jsjs1209@g.skku.edu)

Sungkyunkwan University
2066, Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do, Republic of Korea

December 10, 2023

Abstract

In this report, we will demonstrate a new approach combining Chat-GPT and Retrieval Augmented Generation. Our goal is to increase the accessibility of youth policies to people in Korea. This access will allow young people to more easily obtain policy information, determine eligibility, and apply quickly. As a demo, we created a web application that receives the user's personal information and recommends policies. These attempts will increase the accessibility of youth policies and provide many opportunities to young people who do not receive various benefits. Additionally, in an environment limited to the metropolitan and Gyeonggi areas, it can be applied to many policies across the country and around the world.

Keywords— A I, Chat-GPT, Youth Policy, Retrieval Augmented Generation

1 Introduction

A substantial portion of the budget is allocated to welfare and housing programs for young individuals, and this budget is gradually increasing. However, a significant challenge lies in the fact that many people are either unaware of the existence of youth policies or, even if they are aware, struggle to identify the policy that best aligns with their needs, consequently missing out on vital benefits. These challenges are compounded by the dispersion of policy information across multiple websites, making it arduous to access comprehensive and pertinent details. The barriers to accessing these policies often appear insurmountable.

Recognizing the pressing necessity to bridge this information gap and ensure broader access to entitled benefits, there is a critical demand for services that can streamline and present youth-oriented policies in a user-friendly manner. Moreover, the potential to dramatically simplify the process arises if government-supported policies, tailored to individual circumstances, can be intelligently recommended based on personal information. This convergence of challenges and opportunities has catalyzed the search for innovative solutions.

In the era of advancing technology, we are witnessing the emergence of various chatbots employing Retrieval Augmented Generation(RAG)[1], offering a promising avenue for improved service delivery. Within this context, the concept of harnessing the capabilities of these chatbots to provide more accessible and personalized government support services has taken shape.

Furthermore, in addition to the chatbot function, our plans extend to various domains, including education, cultural exchange, leisure activities, and, of course, youth policy. In summary, our vision entails creating a community for the younger generation and presenting a unique platform tailored to the needs of young people.

2 Design

2.1 Overall Architecture

The architecture integrates a user-centric front-end with a powerful React and JavaScript framework, ensuring an appealing and dynamic user interface. The FastAPI-powered back-end facilitates efficient RESTful APIs,

database management, and server deployment. The data pipeline, implemented with Python and MariaDB, ensures the availability and preprocessing of relevant data. The machine learning model, based on RAG, provides a robust chatbot with the ability to understand user queries, assess policies, and adapt to changing information. This comprehensive architecture reflects a harmonious integration of front-end, back-end, data pipeline, and machine learning components, setting a strong foundation for a dynamic and user-friendly web application.

2.2 Front-end

In the initial stages of the project, we prioritized user experience and utilized Figma to develop a visually appealing web application. Figma proved advantageous for collaboration, allowing real-time sharing and collaborative editing of designs. To implement the designs finalized in Figma into web pages, we employed CSS. CSS, being a powerful styling language, was essential for adjusting the layout and design of the web pages. For frontend development of the web application, we utilized React and JavaScript. React's component-based architecture enhanced code reusability, thereby improving development productivity. Additionally, JavaScript was employed to achieve dynamic user interfaces and implement efficient data communication with the server.

Figma, with its collaborative and real-time editing capabilities on the browser, proved to be a convenient tool. Working concurrently with team members minimized confusion and allowed for swift incorporation of user feedback. Leveraging Figma's prototyping feature, we simulated user paths and flows, enhancing the practicality of the design. The simplicity of design was crucial in Figma, aiming to make it intuitive for users. To address these challenges, the team actively discussed and researched various references, ultimately contributing to delivering a user-centric, intuitive, and attractive UI.

The choice of React for the frontend development of the project was primarily driven by performance and reusability considerations. React's virtual DOM technology enabled efficient UI updates, and the component-based architecture facilitated modularization and reusability, making maintenance more manageable. The vibrant React community ensured access to various libraries and support. JavaScript, being a powerful scripting language executed in browsers, aligned well with the project's requirements. Its asynchronous processing was essential for efficient data communication and implementing the dynamic nature of the web application. Additionally, the highly scalable nature of JavaScript provided compatibility across various browsers, ensuring a consistent experience for users in diverse environments.

However, handling asynchronous tasks and managing data requests posed challenges, leading to issues such as data inconsistencies and prolonged loading times. This was addressed by implementing async/await for asynchronous tasks and utilizing libraries like Axios or the Fetch API for effective data management. The complexity of implementing routing and navigation was another challenge, resolved using libraries like React Router for routing management and optimizing performance through code splitting when necessary.

2.3 Back-end

We needed the Python back-end framework to run the AI Python model, and we used FastAPI. FastAPI has the advantage of providing very fast speed, being intuitive, and having fewer bugs. For the server, we applied the FastAPI to create RESTful APIs based on the required functionalities. By providing RESTful APIs, we have gained scalability, reusability, and ease of maintenance.

We used MariaDB DBMS to manage policy data and user data needed for the model. For the database, we deployed SQLAlchemy as ORM to establish a connection with the MariaDB database. This approach made it easy for us to manage the database, and ORM made it unnecessary to create SQL queries separately. The Figure1 allows you to check the Entity-Relationship Diagram about the User. I created User Info for the User and envisioned an entity that can store folders and contents of folders by the user.

For the server, an instance was created using AWS EC2 and distribution was planned using NGINX.

2.4 Data Pipeline

We build our data pipeline by Python, MariaDB, DDL(Data Definition Language) with SQL. Since a data pipeline is a fundamental and important component in a data-driven service, an efficient data pipeline should be necessary. We build a data pipeline within five steps. EDA is the first step of our data pipeline, beginning with understanding the structure of the data. The EDA step will give a comprehensive understanding of the raw dataset and guide further steps in the data pipeline. We decided to get our data from https://www.youthcenter.go.kr/main.do. The second step is the Implementation of the crawling code. We used python

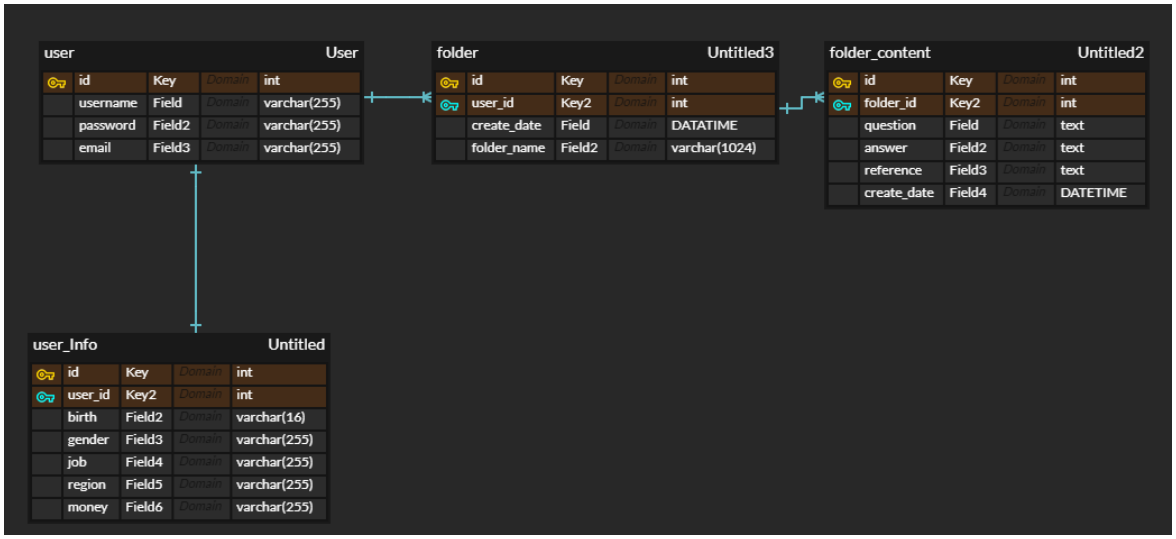

Figure 1: User Database

for implementing web crawling code. we need to get some necessary libraries and modules that will be used for web crawling. The third step is data preprocessing. Since our crawled data, text is for humans, not for our models. we should preprocess our data to fit with our models. After preprocessing, we need to handle it with databases. we made our databases with mariaDB and DDL with SQL. The final step is automation & monitoring. We use youth policy data for our service, this means that data is volatile, automatic crawler is necessary. In addition, we implemented a monitoring code for our service, which can be implemented in the EC2 server.

2.5 Machine Learning Model

Our chatbot should include the following three functions to understand the user's question intention and recommend appropriate policies. First, it should be able to understand the user's questions and generate natural answers. Second, it should be able to judge the policies that users need. Third, it should be possible to check periodically changing policy information and reflect it in generating answers.

We utilize RAG to achieve these functions. RAG is an NLP technique that intertwines the robustness of pre-trained language models with the capability to retrieve and utilize external knowledge sources. The RAG pipeline can be divided into two parts: the retrieval part that searches for information related to the given question and the generation part that generates answers with the retrieved information. During the retrieval part, relevant documents or text snippets are fetched from an external knowledge source based on the given question. The knowledge source is a storage where information that the model wants to refer to is stored. Subsequently, the retrieved context is used to generate a response for the input in the generation part. Since the generative model in the generation part receives the question and the information necessary to answer the question, it is possible to generate an appropriate answer to the question by referring to this information. In general, the generative model is a large language model (LLM) that has the ability to generate a natural answer to a user's question.

RAG has the following advantages over fine-tuning, a traditional NLP technique that reflects new knowledge in generative models.

- It is easy to make the model answer information that has not been trained beforehand. The chatbot we build should periodically reflect the removed or newly created policies. In the case of the Fine-tuning methodology, the model must be newly trained each time the policy is updated, and because LLM is huge, overhead occurs severely. On the other hand, in the case of RAG, only the information from the external knowledge source needs to be updated without additional LLM training, resulting in relatively very little overhead.

- Catastrophic forgetting problems can be prevented. Catastrophic forgetting is the tendency of an artificial neural network to abruptly and drastically forget previously trained information upon training new information. In LLM, a catastrophic forgetting problem can occur when fine-tuning is performed[2]. On the other hand, in the case of RAG, since additional training about LLM is not performed, forgetting existing knowledge does

not occur.

Through the pursuit of these goals, this project will significantly augment the accessibility and efficacy of youth policy, ensuring that young individuals have access to accurate information, engage with supportive communities, and effectively apply for policies that can substantially benefit them.

3 Implementation

3.1 UX/UI

Followings are explanations of the representative User Interface and the corresponding User Experience applied to our service, Chat Pub.

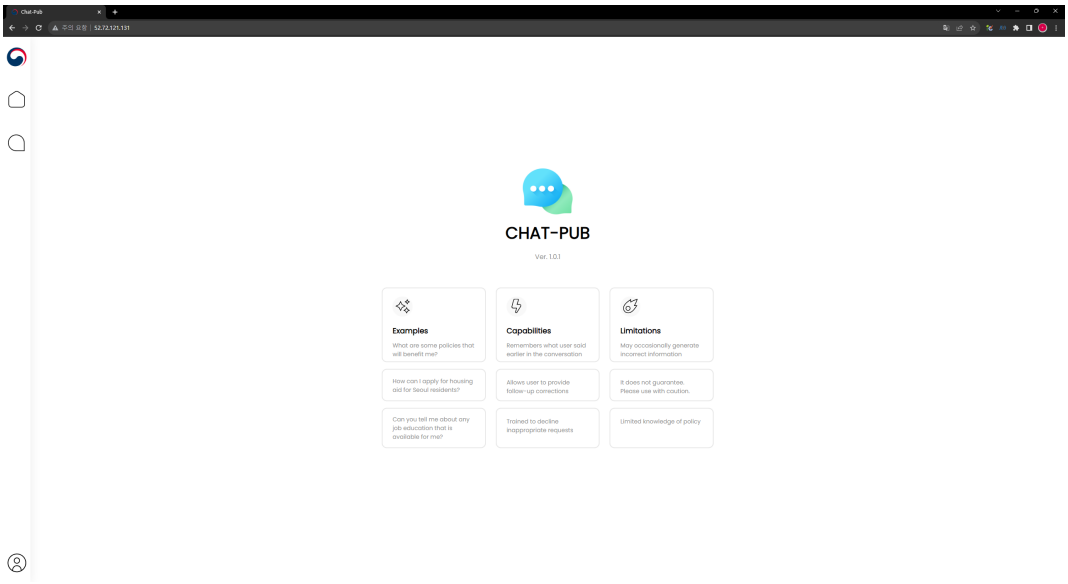

Figure 2: Home Page.

On the Home Page, we offer a brief overview of our service. We provide details on what questions users should ask, what they can and cannot do. Efforts were made to prevent confusion by providing users with a clear understanding of how to use the service through concise explanations. Intuitive menu bar icons on the side allow users to easily find the features they are looking for. Icons are arranged based on the functions they represent, including an icon to navigate to the home page, an icon to enter chat rooms, and icons for logging in and out.
As our service involves obtaining personal information to offer tailored policies to users, logging in is mandatory. Authentication is carried out based on the user ID and password used during registration. Only after successful authentication can users provide additional personal information and access the chat service.
If the login is successful, the user is directed to the Chatroom interface. On the right side, users can create a new chat room through 'New Chat' to initiate a chat with ChatPub. Users are not required to maintain all conversations in a single chat room. Depending on the user's needs, they can create rooms for specific topics such as housing policy discussions, job education policy discussions, and more, allowing them to receive and manage information accordingly.

3.2 Front-end

The Personal Info Page is a page that can be accessed and modified only after authentication through login. Upon login, users are required to provide detailed information that was not entered during the registration process. To assess the suitability for policy support, users need to fill in additional details such as date of birth, gender, location, workplace, and income level. This information ensures more accurate policy recommendations when requesting policy information from Chat Pub in the future. Moreover, it allows Chat Pub to fetch relevant information automatically without users explicitly providing detailed information, enabling personalized policy recommendations. Achieving this functionality required seamless communication with the server. Additionally, it was necessary to promptly receive and display updated data on the web application, as well as to verify that

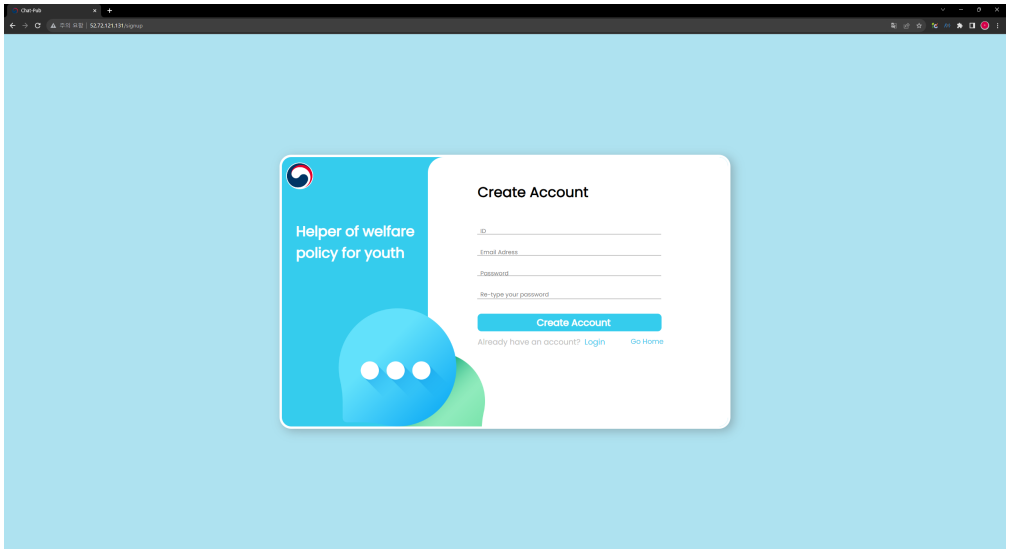

Figure 3: Signup Page.

the provided data was appropriately sent to the database for future searches.

Once authentication is complete and relevant personal information is entered, Chat Pub can provide responses corresponding to user queries. Even if users do not explicitly specify details such as their place of residence, age, or gender in their questions, Chat Pub generates accurate responses based on the provided data. Furthermore, it offers references related to the information, encouraging users to verify and apply for policies. Implementing this feature required distinguishing between user inputs in the chat room and responses generated by Chat Pub. Additionally, to ensure visibility, the retrieved data needed to be sorted.

The front-end implementation emphasizes user interaction, providing a solid foundation for future expansion and enhancement. Ongoing efforts for continuous improvement and the addition of diverse features to enhance user experience are anticipated. We look forward to our web application offering a more efficient and user-friendly environment.

3.3 Back-end

According to ERD, you created a database in MariaDB. The DB automatic design file can easily manage databases, and also developed DB automatic design files that can be properly connected to other environments, and local environments that can be properly connected to other environments. Using ORM, you can store Data to store Data without SQL query, and it was confirmed that data is certainly stored through Communication. Also, using DB Session to create a function that connects to the database, and closes DB only when necessary. Additionally, Through *CRUD.py, it was separately managed by managing DB and stability.

Using the FastAPI Web Framework, I created a RESTful API. According to CRUD, the content that returns information, the content that returns the PST request, the content that you update existing information, and existing information to delete existing information and existing information. You can check the API that was created by FastAPI Docs.

For the implementation of the login function and security, the OAuth2PasswordRequestForm and OAuth2PasswordBear and JWT were used. Using OAuth2, the user's ID and password were requested and saved. By returning and using the Bear type token to the client, security and safety were emphasized through API communication using tokens. Almost all requests contain user information and are sent with the user's authentication token. For JWT, the password is stored in the DB through Hash to increase security.

3.4 Knowledge Source

During the retrieval phase of RAG, relevant documents or text snippets are fetched from an external knowledge source based on the provided question. The knowledge source is constructed using policy data, undergoing preprocessing in the data pipeline mentioned earlier. The crawled data is stored in MariaDB, and as it consists

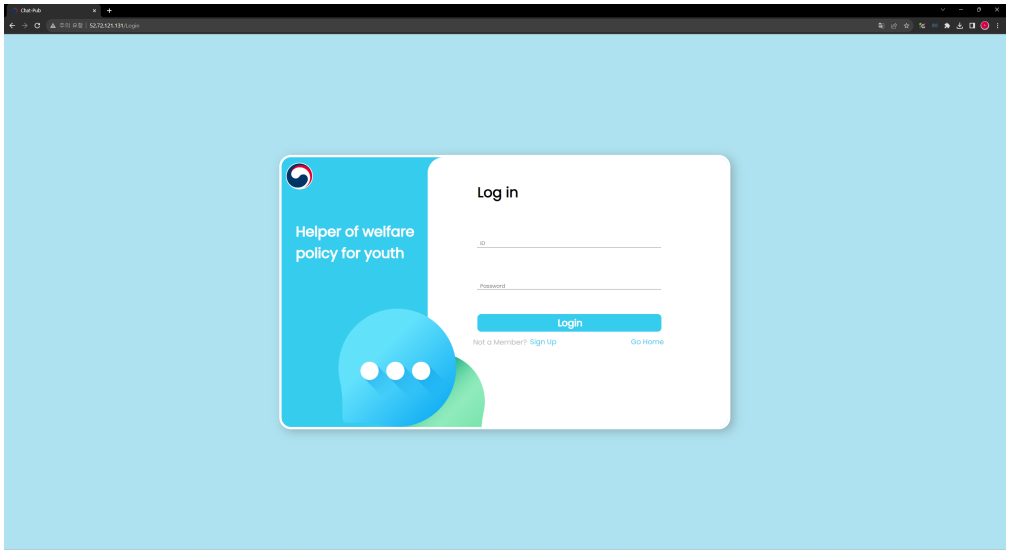

Figure 4: Login Page.

of text data, it needs to be vectorized through a preprocessing step. For this purpose, the Faiss library provided in Python is used. In other words, the knowledge source is the data crawled by an automatic crawler, and this data is transformed into vector embeddings using Faiss. The automatic crawler utilizes libraries such as Selenium for repetitive tasks, Requests for sending web requests, and BeautifulSoup4 for document parsing. Detailed implementation code can be found in our GitHub repository.

3.5 Machine Learning Model

3.5.1 Retrieval Part

The Retrieval Part retrieves information related to a user's question from an external knowledge source. We used an embedding-based retrieval methodology. Embedding means converting a given text into a vector that implies a meaning. The transformed vectors are located close to the vector with similar meanings in the vector space, and information related to a given question can be extracted from the knowledge source through this characteristic.

We used sentence transformers based on Bidirectional Encoder Representations from Transformers (BERT)[3] as an embedding model that performs text embedding. BERT is a model that performs embedding on input text by reflecting context information. After the input text is split into token units, the embedding vector for each token is output by reflecting information on neighboring tokens. Sentence transformers are Python frameworks for state-of-the-art embeddings that apply a pooling layer to the output vector of BERT to create one embedding vector for the entire text. We constructed sentence transformers by adding a pooling layer to the Roberta model pre-trained on the Korean Language Understanding Evaluation (KLUE)[4] data set. The Roberta model is a model that improves the pre-training methodology of the BERT model and has the same structure and input/output as the BERT.

The process of extracting information related to a given query from a knowledge source is as follows. Information in the form of text that exists in the knowledge source is embedded in advance through sentence transformers and stored in the vector database. When a question is entered, the question is embedded through the same sentence transformers and converted into a query vector. The most similar vectors are extracted by performing a similarity operation with the vectors in the vector database. After checking the index of the extracted vectors, the most similar information to the question is extracted by indexing in the storage that stores the information in the form of text.

Sentence transformers are trained so that the vectors of sentences with similar meanings are located close, and unrelated sentences are far away. When two sentences are entered, sentence transformers generate an embedding vector for each sentence. If the given sentences have similar meanings, they are trained so that the cosine similarity value of the two embedding vectors becomes 1. If they have non-similar meanings, they are trained to become 0.

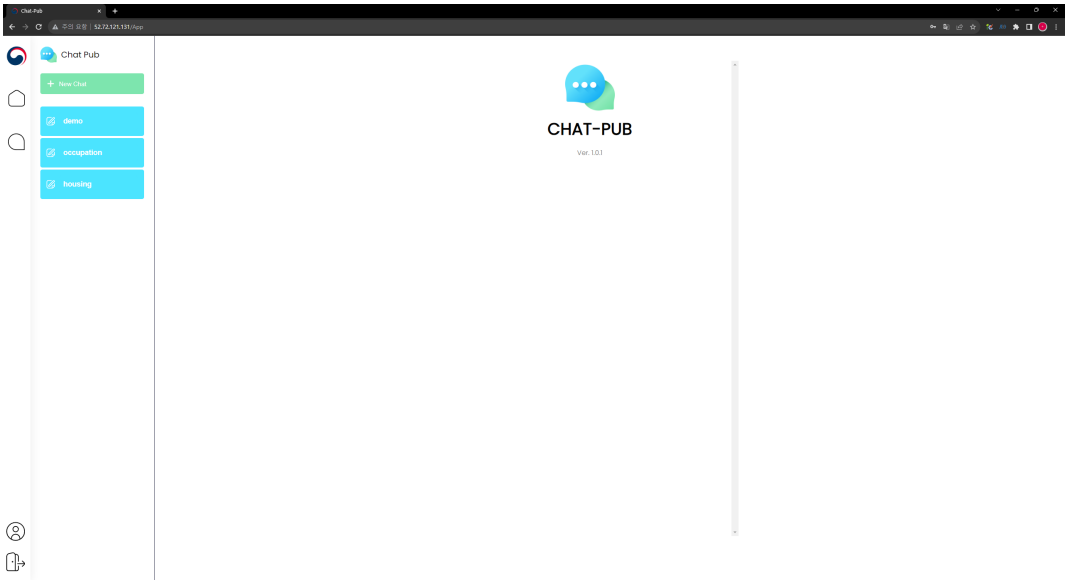

Figure 5: Chatroom Page.

We trained sentence transformers using the in-batch negative sampling methodology[5]. Negative samples, i.e., pairs of sentences that are not similar, are constructed in a batch. In a batch consisting of question and related passage pairs, the i-th question and the i-th passage are considered positive samples, and other data are considered negative samples. Therefore, when the batch size is n, the model is trained so that the cosine similarity of two sentences is 1 for one positive sample and the cosine similarity of 0 for n-1 negative samples.

3.5.2 Generation Part

The generative model answers users' questions using information extracted from the retrieval part. The generative model uses the input passage information to answer a given question and generate a natural form of answer. In addition, if sufficient information for an answer is not provided or a question unrelated to the policy is given, it should be output that the answer cannot be generated.

To meet these functions, we used the gpt-4-turbo model provided by OpenAI API. If an open-source LLM such as Llama[6] is used, a large model should be used to ensure inference performance, which requires a server equipped with a high-quality GPU. Therefore, we used the API in terms of cost and efficiency.

In the training step, it is impossible to directly adjust the model's weight since we utilize API in the generation part. So, we improved the performance through prompt tuning.

3.6 Dataset

With self-built datasets, We fine-tuned the sentence transformers in the retrieval part and evaluated the prompts in the generation part.

Due to the use of API in the generation part, it is difficult to train the retrieval and generation parts at the same time using the same datasets and evaluation metrics because we cannot obtain the training loss, which means the degree of error in the model's training dataset in the generation part. Therefore, we trained and tested the retrieval and generation parts separately.

3.6.1 Retrieval Part

We need a dataset consisting of policy questions and related passage pairs since we use the in-batch negative sampling methodology. We collected passages through the policy site crawling, and question generation was performed using the gpt-3.5-turbo model to generate questions that could be answered through the passages.

Since question generation must simulate user questions in various forms, the types of output questions must

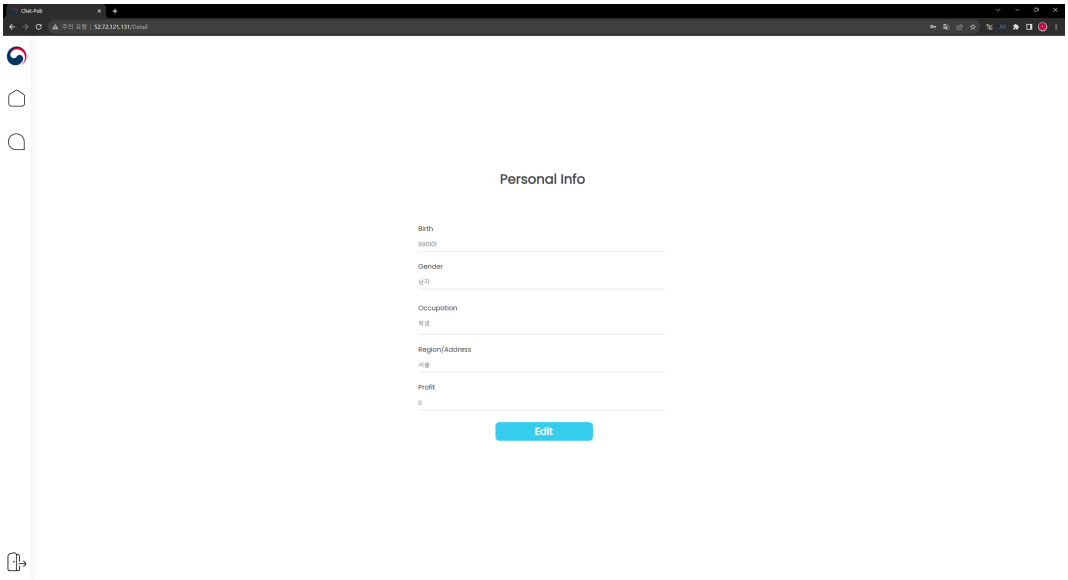

Figure 6: Personal Info Page.

also vary. Therefore, we organized a prompt so that the model performing question generation can generate various types of questions, such as asking questions with little information and asking questions in the informal language (the prompt details are in Appendix A).

To evaluate whether the dataset we built helped improve the performance of the model, we adopted the evaluation methodology presented in the paper 'Generating Datasets with Pre-trained Language Models.'[7] In this paper, the authors used pre-trained gpt to generate the training data for sentence transformers training. Then, they evaluated the benchmark data, often used to evaluate the performance of sentence transformers.

In our setting, we evaluated the model on the KorQuad[8] dataset, which consists of the necessary passage pairs to answer questions and questions, after training with datasets generated through crawling and gpt-3.5-turbo.

3.6.2 Generation Part

Generative model generates answers through user questions and passages extracted through sentence transformers. For prompt tuning, we must be able to evaluate the quality of the answers when we enter our prompts into LLM. We used the LLM as Judge[9] methodology to evaluate whether our prompt induces LLM to produce better answers.

LLM as Judge uses high-performance LLM to evaluate the output results of the target generative models. The output value for the input value is not set, and the judge LLM that evaluates the generative models evaluates the output of the generative models according to the criteria specified in the input prompt. In other words, the prompt to be evaluated is written, the question and passage are input together into the target generative model, and the generated result is inputted to the judge LLM for evaluation. Therefore, the dataset used for prompt evaluation is a question and the related passage pairs, which are composed of 40 randomly extracted from the dataset used when training the sentence transformers.

4 Evaluation

4.1 Retrieval Part

We used Huggingface Transformers and Pytorch Library. We employed the Adam optimizer, adjusting the learning rate from a set of $2 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}$. The warm-up steps were determined as one-tenth of the total training steps, and the weight decay coefficient was set to 0.01. Batch size was chosen from 32, 64, 128, and we conducted 5 epochs with early stopping. The reported score is derived from the optimal hyperparameter configuration, determined by validation loss.

The evaluation results are shown in Figure8. Baseline represents the evaluation results of the pre-trained

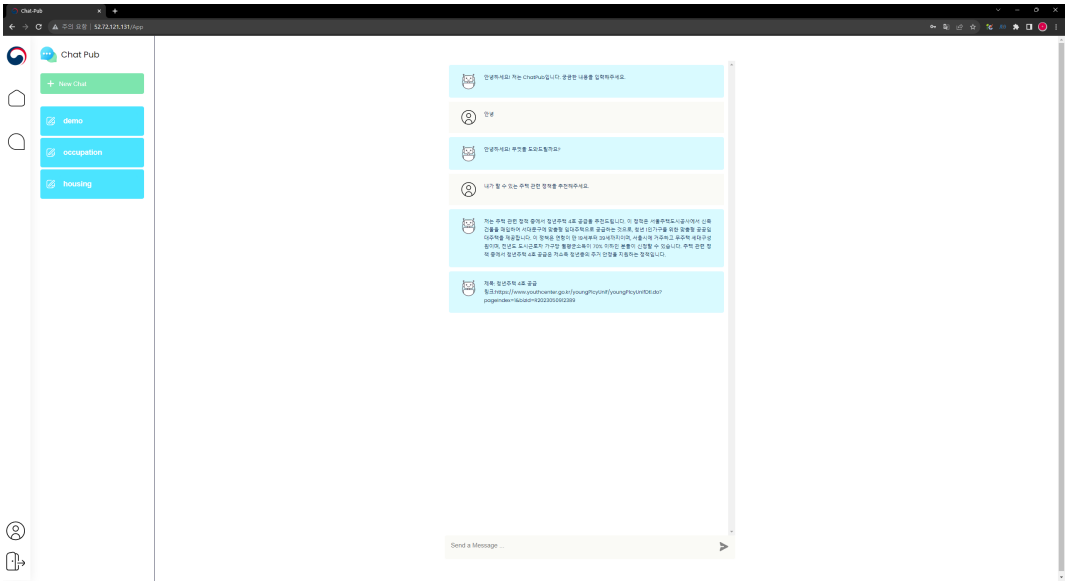

Figure 7: Chat Room with Answers.

model in Korean, Klue-nli represents the evaluation results of the fine-tuned model with data of Korean pairs of similar meanings and Ours represents the evaluation results of the fine-tuned model with the data we generated. Each numerical value represents accuracy on the KorQuad dataset.

As the results show, when fine-tuning with the data we generated, we found that the performance was improved for the evaluation dataset, which means that the data we generated was built to allow the model to perform well in embedding.

	MODEL		
		BERT	**Roberta**
TRAIN DATASET	**Baseline** (pre-trained)	**0.791**	**0.725**
	Klue-nli (pre-trained + fine-tuned)	**0.873**	**0.869**
	Ours (pre-trained + fine-tuned)	**0.888**	**0.888**

Figure 8: Evaluation Result for Sentence Transformers on KorQuad Dataset.

4.2 Generation Part

LLM as Judge considers the generated output of the target models in two main ways: comparison is a method of evaluating which of the output values of the two target models is better, and grading is a method of scoring the output values of one target model from 1 to 10. Both methods rely on LLM's reasoning results rather than statically evaluating them by objective indicators. Judge LLM prompts for comparison and gradation presented in this paper are in Appendix B and C.

Since traditional NLP evaluation methods such as Rouge[10] evaluate according to the keyword-based matching rate between the correct answer label and the output value, it is difficult to assess the natural and free answers of the generative model and to reflect human preference. In addition, if the language expression is different even though it has the same meaning as the correct answer label, there is a limitation in evaluating the generative model because it is considered incorrect. Therefore, we adopted the LLM as Judge methodology to effectively assess the LLM output result from the perspective of human precedence.

We used LangSmith's Hub for prompt exploring, which provides prompts for various tasks. We explored the prompts for retrieval augmented generation. After setting up a few base prompts, we explored the optimal prompts with some modifications.

The final selected prompt is as shown in Figure9. It was customized to suit the characteristics of a Korean policy QA chatbot and showed the best performance in comparison and grading evaluation.

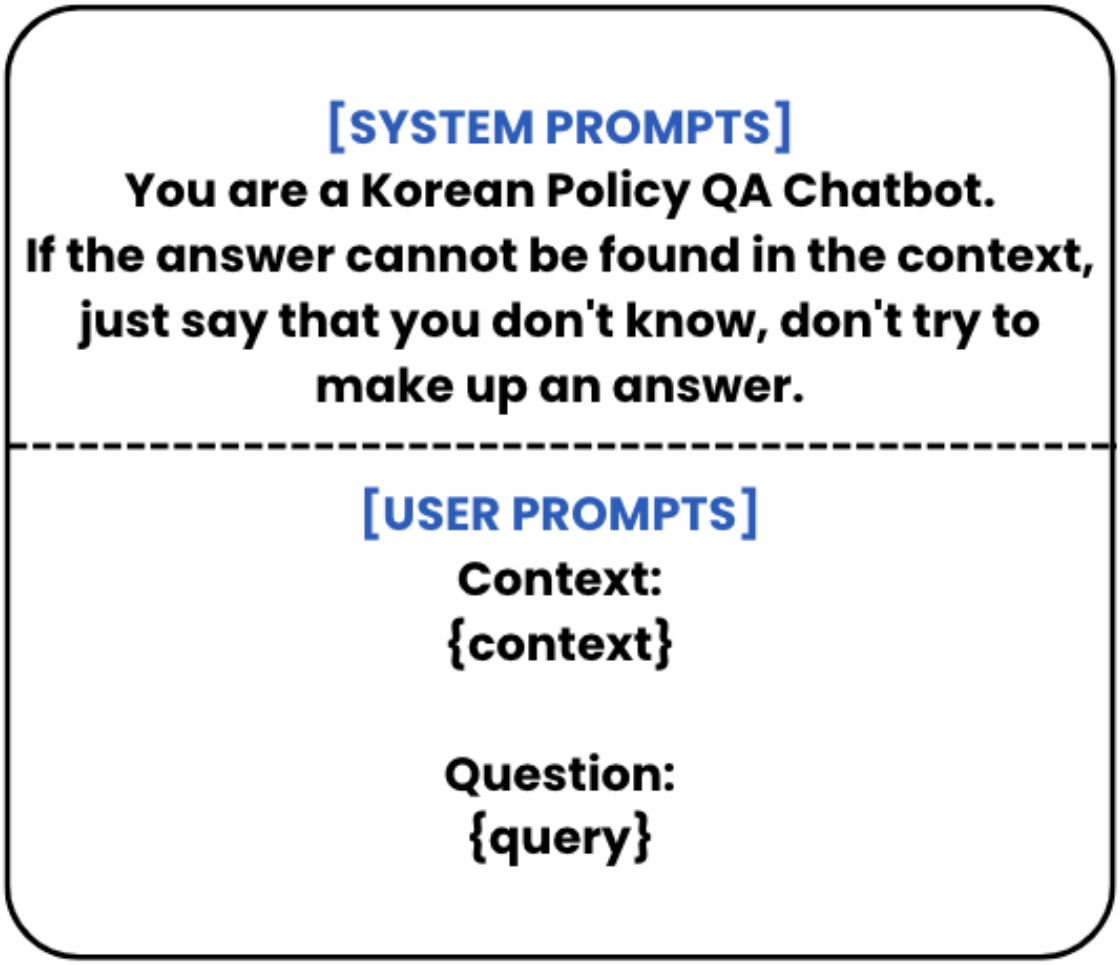

Figure 9: Prompt for Generative Model.

It averaged 7.5 scores in the grading task and performed better than the prompt, which provided much instruction in the comparison task.

5 Limitations and Discussions

In the pursuit of delivering a cutting-edge chatbot service tailored for young individuals, it is imperative to acknowledge and address certain limitations. Firstly, the service's dependency on the accuracy and timeliness of policy data introduces challenges in ensuring the reliability of information. Regular updates and a robust data quality management strategy are pivotal to maintaining the precision of both user and response data, thereby upholding the service's trustworthiness.

Another critical consideration revolves around the diversity of user data, as limited collection efforts may inadvertently introduce bias into the service. To foster inclusivity, we are committed to diversifying data sources and placing a particular emphasis on underrepresented groups, ensuring a more comprehensive and unbiased user experience. Additionally, the handling of personal information necessitates a stringent commitment to data security and user privacy. Our adherence to strict security policies, alignment with regulatory frameworks such as GDPR, and ongoing security audits serve as cornerstones to safeguard user information and mitigate potential security vulnerabilities. Through these proactive measures, we aim to fortify the service against inherent limitations, ensuring its continued evolution as a user-centric, reliable, and ethically sound platform.

6 Related Work

6.1 BERT

BERT is a state-of-the-art natural language processing (NLP) model introduced by Google in 2018. It represents a significant advancement in the field of pre-trained language representations, as it employs a bidirectional context to better understand the meaning of words in a sentence. The key innovation of BERT lies in its ability to capture contextual information from both the left and right sides of a word in a sentence. Traditional NLP models, such as those based on recurrent neural networks process words in a sequential or unidirectional manner. BERT, on the other hand, utilizes the Transformer architecture, allowing it to consider the entire context of a word by considering both its preceding and following words simultaneously. The bidirectional contextual embeddings produced by BERT have proven to be highly effective in capturing semantic relationships and understanding the context in which words appear. This has led to BERT achieving state-of-the-art performance on various NLP benchmarks and tasks. Researchers and practitioners widely adopt BERT or its variants as a foundational component for many natural language understanding applications, making it a crucial reference point in the field of NLP.

6.2 Sentence Transformers

SentenceTransformers is a Python framework for state-of-the-art sentence, text, and image embeddings. These models aim to capture the semantic meaning and relationships between sentences, allowing for a more effective and nuanced understanding of text data. Unlike traditional models that primarily operate at the word level, Sentence Transformers take into account the entire sentence and its context. One notable approach to Sentence Transformers involves leveraging pre-trained models, often based on transformer architectures like BERT. These models are initially trained on large corpora to train general language patterns and contextual relationships. Subsequently, the pre-trained models can be fine-tuned on specific downstream tasks or datasets to tailor their capabilities to particular applications. The use of Sentence Transformers has demonstrated significant success in various NLP tasks, such as semantic textual similarity and document retrieval. These models excel in capturing the semantic meaning of sentences and are particularly valuable in scenarios where understanding the contextual relationship between pieces of text is crucial.

6.3 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is a natural language processing methodology that integrates both retrieval and generation mechanisms to enhance the performance of language models in various tasks. This approach combines the strengths of information retrieval systems, which retrieve relevant information from a given knowledge base, with the generative capabilities of language models that can create coherent and contextually relevant responses. The basic idea behind Retrieval Augmented Generation is to first retrieve relevant information or context from a large knowledge base or corpus and then use this retrieved information to guide the generative process. This two-step approach helps address the limitations of purely generative models, such as their tendency to produce information that may be inaccurate or inconsistent. RAG has found applications in various NLP tasks, including question-answering and conversational agents. By incorporating retrieval mechanisms, these models can leverage external knowledge sources to enhance their understanding and generate more informed and contextually appropriate responses.

7 Conclusion

In conclusion, our paper introduces a novel approach to enhancing the accessibility of youth policies through RAG. With the overarching goal of empowering young individuals in Korea to easily access, understand, and apply for pertinent policies, we developed a web application that leverages personal information to intelligently recommend tailored policies. This innovative solution not only addresses the challenges posed by the dispersed nature of policy information but also strives to bridge the gap for those unaware of available benefits.

The significance of our work lies in its potential to revolutionize the way young people interact with and benefit from government-supported policies. By combining the capabilities of advanced chatbots with RAG methodology, we aim to not only simplify the policy identification process but also contribute to the broader goal of creating a more inclusive and supportive environment for the younger generation. The application of our approach is not limited to specific regions, offering scalability and adaptability for implementation in various policies across Korea and even globally.

Looking ahead, our vision extends beyond the realm of youth policies, encompassing education, cultural exchange, leisure activities, and community-building for the younger generation. We believe that our unique platform holds the potential to serve as a comprehensive hub for diverse services tailored to the evolving needs of young individuals. In essence, our work represents a forward-looking endeavor at the intersection of technology, government support, and community building, contributing to a more accessible and enriched future for the youth.

8 Appendix

A Question Generation Prompts

<SYSTEM PROMPTS> You are an expert in question generation. Your task is, when you receive information about Korea's youth policy, generate questions that can deduce the answer from the policy information.
Example :

예시 1: "지원내용: 미취업 청년 고른 취업기회 제공과 역량강화 비용 지원\n" Example for output : Q : 내가 취업을 못한 상태인데 지원받을 정책이 있을까?

예시 2: "지원내용: 미취업 청년 고른 취업기회 제공과 역량강화 비용 지원\n" Example for output : Q : 현재 미취업상태 청년들에게 해당되는 혜택이 있는지 알려주세요.

예시 3: "제목: 미취업청년 어학자격시험 응시료 지원사업\n신청 절차: 신청일 당시 도내 주민등록지에 신청\n" Example for output : Q : 어떤 절차를 거쳐야 어학자격시험 응시료를 지원 받을수 있을까?

예시 4: "제목: 미취업청년 어학자격시험 응시료 지원사업\n신청 절차: 신청일 당시 도내 주민등록지에 신청\n" Example for output : Q : 어학자격시험 응시료를 지원받는 방법에 대해 알려주라.

예시 5: "제목: 미취업청년 어학자격시험 응시료 지원사업\n연령: 만 19세 34세\n거주지 및 소득: 도내 거주 미취업 청년(만19 34세)\n- (연령기준일) 1988. 1. 2. 2004. 1. 1.(2023년 1월 1일 기준 만34세 19세)\n※ 지원 연령의 경우는 시군 조례에 따라 연령을 달리 정하고 있음\n - (요건) 응시일 기준 미취업, 지원년도 1. 1.부터 신청일까지 경기도 거주\n학력: 제한없음\n전공: 제한없음\n취업 상태: 미취업자\n특화 분야: 제한없음\n참여 제한 대상: (참여시군) 30개 시군\n※ 성남시(자체추진)\n※ 23년 1차(5월) 신청은 사업 신청일 전 예산확보 된 시군부터 개시\n" Example for output : Q : 나 30살인데 어학시험 응시료 지원 정책 받을수 있어?

예시 6: "제목: 미취업청년 어학자격시험 응시료 지원사업\n연령: 만 19세 34세\n거주지 및 소득: 도내 거주 미취업 청년(만19 34세)\n- (연령기준일) 1988. 1. 2. 2004. 1. 1.(2023년 1월 1일 기준 만34세 19세)\n※ 지원 연령의 경우는 시군 조례에 따라 연령을 달리 정하고 있음\n - (요건) 응시일 기준 미취업, 지원년도 1. 1.부터 신청일까지 경기도 거주\n학력: 제한없음\n전공: 제한없음\n취업 상태: 미취업자\n특화 분야: 제한없음\n참여 제한 대상: (참여시군) 30개 시군\n※ 성남시(자체추진)\n※ 23년 1차(5월) 신청은 사업 신청일 전 예산확보 된 시군부터 개시\n" Example for output : Q : 취업을 못하고 성남시 사는데 어학시험 지원 받을수 있을까?

예시 7: "제목: 미취업청년 어학자격시험 응시료 지원사업\n 연령: 만 19세 34세\n거주지 및 소득: 도내 거주 미취업 청년(만19 34세)\n- (연령기준일) 1988. 1. 2. 2004. 1. 1.(2023년 1월 1일 기준 만34세 19세)\n※ 지원 연령의 경우는 시군 조례에 따라 연령을 달리 정하고 있음\n - (요건) 응시일 기준 미취업, 지원년도 1. 1.부터 신청일까지 경기도 거주\n학력: 제한없음\n전공: 제한없음\n취업 상태: 미취업자\n특화 분야: 제한없음\n참여 제한 대상: (참여시군) 30개 시군\n※ 성남시(자체추진)\n※ 23년 1차(5월) 신청은 사업 신청일 전 예산확보 된 시군부터

개시\n" Example for output : Q : 야 나 취업해야되는데 정책 괜찮은거 있어?

예시 8: "제목: 미취업청년 어학자격시험 응시료 지원사업\n연령: 만 19세 34세\n거주지 및 소득: 도내 거주 미취업 청년(만19 34세)\n- (연령기준일) 1988. 1. 2. 2004. 1. 1.(2023년 1월 1일 기준 만34세 19세)\n※ 지원 연령의 경우는 시군 조례에 따라 연령을 달리 정하고 있음\n - (요건) 응시일 기준 미취업, 지원년도 1. 1.부터 신청일까지 경기도 거주\n학력: 제한없음\n전공: 제한없음\n취업 상태: 미취업자\n특화 분야: 제한없음\n참여 제한 대상: (참여시군) 30개 시군\n※ 성남시(자체추진)\n※ 23년 1차(5월) 신청은 사업 신청일 전 예산확보 된 시군부터 개시\n" Example for output : Q : 제게 좋은 정책을 추천해주세요.
<USER PROMPTS> Make one question using the policy information given below 정책 정보: "passage"

B Judge LLM Comparison Prompts

<SYSTEM PROMPTS> Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below. Your evaluation should consider correctness and helpfulness. You will be given a reference answer, assistant A's answer, and assistant B's answer. Your job is to evaluate which assistant's answer is better. Begin your evaluation by comparing both assistants' answers with the reference answer. Identify and correct any mistakes. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[User Question]
question
[The Start of Reference Answer]
answer_ref
[The End of Reference Answer]
[The Start of Assistant A's Answer]
answer_a
[The End of Assistant A's Answer]
[The Start of Assistant B's Answer]
answer_b
[The End of Assistant B's Answer]

C Judge LLM Grading Prompts

<SYSTEM PROMPTS> Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Begin your evaluation by providing a short explanation. Be as objective as possible. After providing your explanation, please rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

[Question]
question
[The Start of Assistant's Answer]
answer
[The End of Assistant's Answer]

References

[1] Ethan Perez Aleksandra Piktus Fabio Petroni Vladimir Karpukhin Naman Goyal Heinrich Küttler et al. Lewis, Patrick. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv, (2020). Accessed December 8, 2023. /abs/2005.11401.*, 2020.

[2] Zhen Yang Fandong Meng Yafu Li Jie Zhou Luo, Yun and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *ArXiv, (2023). Accessed December 8, 2023. /abs/2308.08747.*, 2023.

[3] Ming Chang Kenton Lee Devlin, Jacob and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv, (2018). Accessed December 9, 2023. /abs/1810.0480*, 2018.

[4] Jihyung Moon Sungdong Kim Won I. Cho Jiyoon Han Jangwon Park Chisung Song et al. Park, Sungjoon. Klue: Korean language understanding evaluation. *ArXiv, (2021). Accessed December 9, 2023. /abs/2105.0*, 2021.

[5] Gogoulou E. Ylipää E. Cuba Gyllensten A. Sahlgren M. Carlsson, F. Semantic re-tuning with contrastive tension. presented at the international conference on learning representations. *(2021). , 2021. Retrieved from https://urn.kb.se/resolve?urn=urn:nbn:se:ri:diva-59816.*, 2021.

[6] Thibaut Lavril Gautier Izacard Xavier Martinet Marie Lachaux Timothée Lacroix Baptiste Rozière et al. Touvron, Hugo. Llama: Open and efficient foundation language models. *ArXiv, (2023). Accessed December 9, 2023. /abs/2302.13971.*, 2023.

[7] Timo Schick and Hinrich Schütze. Generating datasets with pretrained language models. *ArXiv, (2021). Accessed December 9, 2023. /abs/2104.07540.*, 2021.

[8] Myungji Kim Lim, Seungyoung and Jooyoul Lee. Korquad1.0: Korean qa dataset for machine reading comprehension. *ArXiv, (2019). Accessed December 9, 2023. /abs/1909.07005.*, 2019.

[9] Wei Chiang Ying Sheng Siyuan Zhuang Zhanghao Wu Yonghao Zhuang Zi Lin et al. Zheng, Lianmin. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv, (2023). Accessed December 9, 2023. /abs/2306.05685.*, 2023.

[10] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *In Text Summarization Branches Out, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.*, 2004.